16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

A SYSTEM ON CHIP DEDICATED TO PIPELINE NEIGHBORHOOD PROCESSING FOR MATHEMATICAL MORPHOLOGY

Christophe Clienti, Serge Beucher, Michel Bilodeau

Center of Mathematical Morphology, ENSMP
35 Rue Saint Honoré, 77300, Fontainebleau, France
phone: + (33) 1 64 69 47 06, fax: + (33) 1 64 69 47 07, email: christophe.clienti@ensmp.fr
http://cmm.ensmp.fr

ABSTRACT

This paper describes a system on chip for image processing. It is based on a pipe-line of neighborhood processors named SPoC and is controlled by a general purpose processor. Each SPoC are connected one to the other through a reconfigurable data path to get more adaptability and their structure exploits temporal and spatial parallelism to speed up computations and minimize memory transfers. Two applications, a motion detection algorithm and a licence plate extraction, are presented to show performances in terms of speed, embeddability and re-usability of the SoC. Comparisons with many architectures such as digital signal processors, workstations or embedded SIMD processors are made to benchmark the platform and prove the originality and the strength of our solution.

1. INTRODUCTION

For more than 40 years, the use of Mathematical Morphology for image processing has been constantly growing in many domains such as medical imaging, computer vision, multimedia application or security [1], [2].

Faced with an increasing demand for efficient image processing on embedded systems, embedded traditional processors are not powerful enough to target real-time constraints required by applications. Moreover applications are more and more sophisticated and dedicated architectures for image processing are often only specialized in peculiar parts of them. We can cite many architectures only optimized for some basic operations [3], [4].

Digital signal processors and SIMD processors are not always well suited to very large numbers of calculations required by some sequential stages of an application. Moreover sensor applications require a lot of memory due to the increasing resolution and due to algorithms increasing complexity. Processors cannot be fast enough and cache misses may occur [5]. Wide VLIW or Systolic architecture composed by hundreds of simple processing elements are an alternative to compute algorithm rapidly, however programming model may be very complicated [6].

Using neighborhood operators in a pipeline way - where each stage could map an application part - is a possibility to simplify processing and improve performance. In the past, such pipelines were built off-chip with many circuits arranged together to target the performance required [7]. Mathematical Morphology is really well suited to this kind of architecture because it is based on simple operations which can fit into one or more stages of a pipeline. Today we have the opportunity to create reconfigurable architecture with many on-chip neighborhood operators interfaced with a generic purpose processor to schedule processing.

The first part of the paper is dedicated to a short reminder about Mathematical Morphology. In the second part, the platform architecture is presented and emphasis is being placed on vectorized neighborhood processors able to extract multiple sub-windows at a time. In the third part, two applications, a video survey motion detection and a licence plate extraction, are described and a benchmark is done between our SoC and different platforms. Finally, the last part is dedicated to improvements and future works.

2. MATHEMATICAL MORPHOLOGY REMINDER

Mathematical Morphology was created in late 60's by G. Matheron and J. Serra. The two morphological basic operations are dilation and erosion, from which, all others operations can be composed.

An Erosion ε of an function f by a flat structuring element B is defined as follow:

$$[\varepsilon_B(f)](x) = \min_{b \in B}(f(x+b))$$

The dual operator is the dilation δ:

$$[\delta_B(f)](x) = \max_{b \in B}(f(x+b))$$

A structuring element is a neighborhood window, where for each pixel of the image a neighborhood configuration is considered.

Those primitives are considered as basic tools achieving a lot of more complex operations :

- gradient : $g = \delta - \varepsilon$
- opening : $\gamma(f) = \delta(\varepsilon(f))$
- closing : $\varphi = \varepsilon(\delta(f))$
- alternate filters (af) : opening and closing (vice versa)
- alternate sequential filters (asf) : size increasing successive openings and closings (vice versa)
- geodesic transformations
- distance, quasi-distance
- skeleton (using Hit or Miss operators)
- watershed
- ...

An exhaustive introduction to Mathematical Morphology can be found in [2].

The challenge is to build the best optimized architecture to compute rapidly erosion/dilation and to plan them efficiently.

3. PLATFORM DESCRIPTION

The aim of the designed circuit is to compute Mathematical Morphology algorithms in the framework of embeddable systems. It is able to acquire images from a source, compute them, and transmit results in real-time constraint evaluated at 30 frames per second. The functional view of the SoC is presented in figure 1.

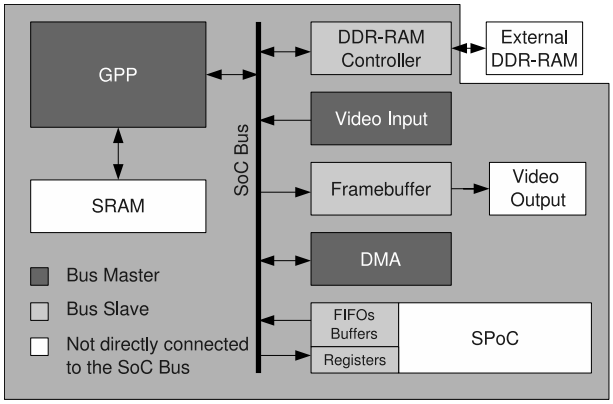

Figure 1: System on Chip for Image Processing With Mathematical Morphology

3.1 Chip structure

The circuit is composed of a simple General Purpose Processor (GPP) which is devoted to control algorithm scheduling and communication in and out of the circuit. If necessary, the GPP can access image pixels located in the DDR-RAM. This must be sporadic because of an important memory latency when cache misses occur. The GPP is a simple 32 bits RISC processor operating at a low frequency and may run an Operating System to manage network communications. Image processing are deported in a dedicated part of the circuit named SPoC (Several neighborhood Processors On Chip). In order to feed the pipeline with pixels through FIFOs, a standard DMA is used to optimize the memory transfer from and to the DDR-RAM.

The input video system supports standard video inputs as PAL or NTSC, converted into numerical trams with an external ADC. The video input logic inside the SoC is used to decode frames and to take control of the SoC bus to send pixels directly in DDR-RAM. Optional parts are the framebuffer and the video output which represent a basic graphic system mainly used to display some image processing results.

SPoC is optimized to minimize image memory transfers between each application stage. The pipeline structure allows to process some Mathematical Morphology operations with only one image load/store and it exploits temporal and spatial parallelism to speed up computations. SPoC processing configuration is built with a list of basic operations parameters to be executed such as dilation or erosion, and with a data path parametrization. All these parameters must be stored in configuration registers before starting a process. Border side effects are automatically managed, and only tasks devoted to the GPP are SPoC register bank programming and DMA configuration.

Once the pipeline is configured, images pixels are sent to input FIFOs and the process can start. If two images must be sent to the accelerator, the process will start only when pixels will be available in each FIFO. After a latency equal to some lines of an image, depending of the pipeline depth, pixel results are produced and stored in output FIFO at each clock cycle. The GPP must program the DMA to store pixel progressive results in main SoC memory.

3.2 SPoC structure

SPoC is composed by many neighborhood processors, called PoC, associated in a parallel and in a serial way through a reconfigurable data path and a programmable ALU units, as shown in figure 2. One PoC is able to compute erosion, dilatation, gradient and optionally median filter and convolution with kernel up to 3×3.

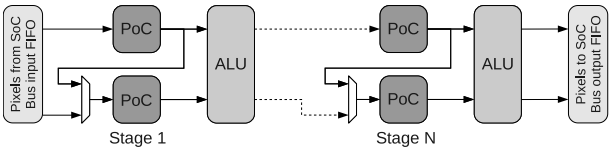

Figure 2: SPoC internal overview

This pipeline structure is really relevant for Mathematical Morphology, and is most often full at each application stage. Moreover SPoC is a MIMD architecture and can process one image in both parallel PoC with the same instruction, or process two different images with different operations. Multiplexers in the pipeline allow to use all PoC in the circuit in a serial way. This option is very interesting when we compute alternate sequential filters until a big structuring element size.

The ALU unit manages results from parallel PoC, and also reconfigure data paths before a process start. This part is important to compute geodesic operation, distance transformation or any complex transformations. The ALU unit embeds also a Constant Pixel Register (CPR) to compute operations on each pixel with a constant value instead of an image. The figure 3 represents the ALU functional view.

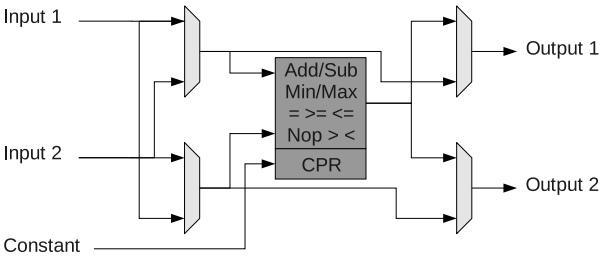

Figure 3: ALU functional overview

The ALU unit allows to compute addition, subtraction, saturated addition, saturated subtraction, absolute value of a subtraction, minimum, maximum on both image inputs or between an input and CPR. The NOP instruction is decomposed in two NOP instructions in order to choose which input must be directly copied to the output. Comparison operations have been introduced to do conditional move or threshold, and many combinations between inputs and CPR are available. For example, we can compare each pixel of two images and replace true values by the CPR when a test is true. We

can also test if a pixel is superior to CPR and set the output to the max value authorized by the pixel bus width (0xFF for 8-bit pixels).

PoC implements a parallel neighborhood extractions to speed-up computation time as shown in figure 4. At each clock cycle, PoC extracts N successive neighborhoods depending on the pixel vector of size N. The figure 5 shows a simplified view of the neighborhood extraction architecture with a vector of size 4 and a 13×6 image. Register RA1 through RF2 are neighborhood registers, and at each clock cycle 4 pixels in a little endian order are set to the PoC entry from inputs FIFO. PoC operators are composed of 2 delay lines PDL1 and PDL2 to correctly extract neighborhoods (figure 4) and they must be reconfigured to match the image width. The size of a delay line is the same as used in a standard neighborhood extractor [7], but the geometry must be modified to comply with the pixel vector size.

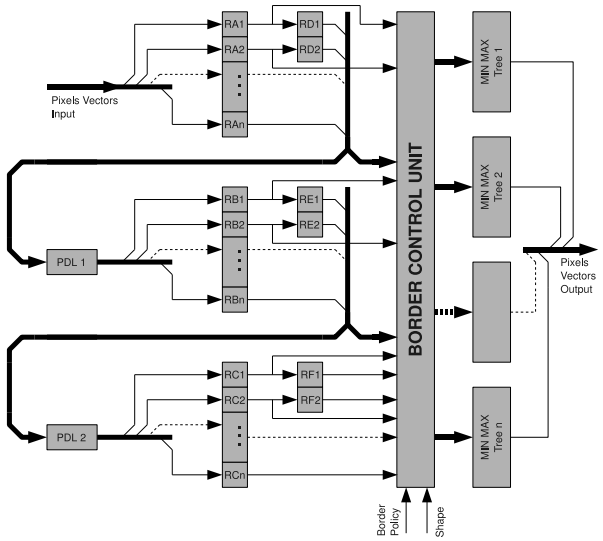

Figure 4: Functional view of PoC

Centers of neighborhoods are represented by registers {RB2,...,RBn,RE1} and for example, the last structuring element extracted, considering a raster scan order in the image, is composed of registers {RAn,RBn,RCn,RD1,RE1,RF1,RD2,RE2,RF2} where n represents the vector size. The covering between last neighborhood at cycle k and first neighborhood at cycle $k+1$ is guaranteed by data paths established, for example, between registers {RA1,RA2} and {RD1,RD2}.

Extracted pixels could be deselected in the Border Control Unit regarding the neighborhood positions in the image and the structuring element shape.

The Border Control Unit is built in three stages. The first stage is a row and column counter, the second one is dedicated to comparators to correctly set or unset pixels in the third stage.

The deselection of registers depends on calculations which must be done with the neighborhood and depends on the structuring element defined by the user (Square, Hexagon, Cross, ...). For an erosion, deselected pixels must be replaced by the maximum value authorized by the pixel bus width. For a dilation, deselected pixels must be replaced by zero. For the median filter, the policy chosen is to replace

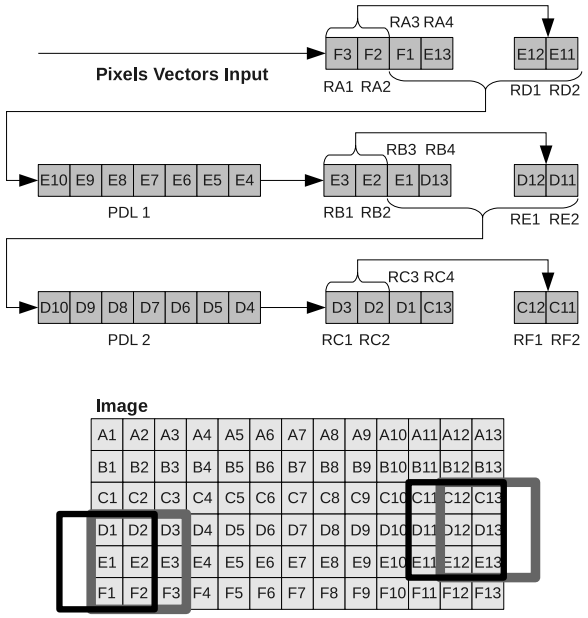

Figure 5: Simplified view of PoC for a vector of 4 pixels

values of neighborhood as the image border was filled by a checkerboard. Figure 6 shows the principle and how the sort works.

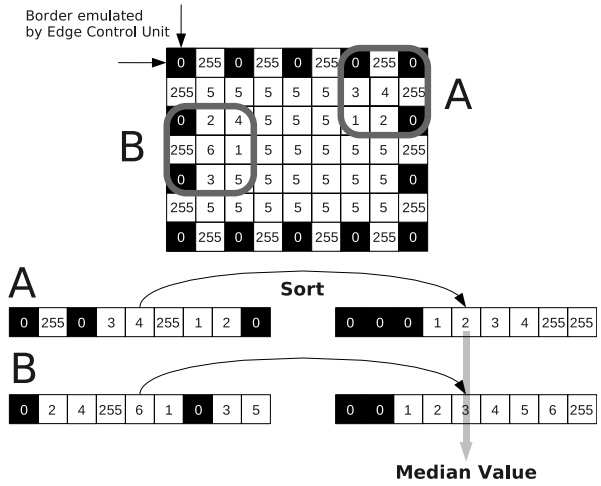

Figure 6: Median Checkboard

Once border sides are correctly processed, pixels go on through Min/Max tree. This arithmetic unit is presented in figure 4 and could be replaced before synthesis step by a multiply and accumulate tree to compute convolution or by a bubble/fusion sort fully pipeline. With such a sort system, we can process every rank filters (erode, dilate and median are special cases of rank filters). Due to some area constraints we decided to synthesize only the first stage of SPoC with Sort Unit instead of Min/Max tree. It is not necessary to add more stages with bubble sort because the composition rules used for erosion and dilation, which allow us to compose big filters, is not valid for others rank filters.

3.3 Programming model

The dedicated architecture could not be addressed directly from video acquisition component because an application could not fit completely into the pipeline and several passes with the same images must be performed. That is the reason why the pipeline is plugged to the SoC bus to have the opportunity to process several times images in the pipeline. It is able to process multiple pixels at each clock cycle but the general purpose processor in the SoC is not fast enough to feed SPoC with pixels. The processor must just configure DMA to load or store data in FIFO which are written or read by the pipeline. The programming step are shown in figure 7.

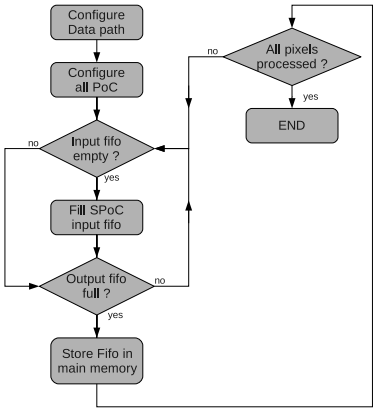

Figure 7: Programming Model

4. APPLICATIONS AND BENCHMARK

The platform presented in this paper aims at running many embedded morphological image processing applications and two examples are used to benchmark the platform. The first example is a standard video survey motion detection and the second one is an application to extract licence plate. Platforms used for benchmarks are:

- Intel Pentium IV running at 3 GHz with 1Mo cache L2 and using SSE2 instructions
- Trimedia DSP running at 320 MHz
- Storm-1 SP16HP-G220 running at 700 Mhz [6]

One SPoC stage use approximately 60k asic gates without taking account line memories. The quantity of memory needed in a single stage corresponds to four double port memories. The size of a double port memory is equal to image line width considering also the pixel bit length. Typically, for 8-bits per pixels and 512 pixels per row, a single SPoC stage needs 2 ko memory.

Our SoC is routed with a 8-stage SPoC pipeline and running at 100 MHz in a Xilinx FPGA Virtex4 LX 60. It occupies about 21000 slices and use 64 ko DPRAM, 32 for the GPP and 32 for the dedicated pipeline which processes 1024×1024 8 bits per pixel images. SPoC pipeline could be synthesized up to 300 Mhz with Virtex 4 FPGA and up to 400 MHz with Virtex 5.

4.1 Peak computational power

First of all a benchmark is carried out to measure, with basic mathematical morphology operations, the peak computational power of our architecture against a Pentium 4, a Trimedia and an Storm-1 vision chip. We consider here 512×512 8 bits per pixel images, and benchmark operations are size increasing erosions.

Op. \ Arch.	P4	Trimedia	Storm-1	Our SoC
erosion 1	0.8	8.2	0.06	0.65
erosion 3	2.5	24	0.18	0.65
erosion 10	8.3	82	0.60	0.65
erosion 16	8.3	82	0.96	0.65
erosion 17	14	140	1.03	1.31

Figure 8: computational time for different erosion of 512×512 images (in ms)

Figure 8 shows some timing for erosion of size 1 to 17. A size one erosion corresponds to an erosion 3x3 in 8-connexity. The 6-connexity was also used but timings are almost the same. Erosions are iterated many times to build bigger structuring elements (composition rule in mathematical morphology). We notice for standard processors that the timing grows at each erosion iteration, while the timing for our SoC grows only when a second pass is needed. It is worth mentioning that in a 8-stage SPoC, we can map up to 16 erosions. A seventeenth erosion could not be mapped in the first pass so a second one is mandatory. The challenge for an application is to use as much as possible the whole SPoC pipeline.

4.2 Motion Detection Algorithm

The method used to extract motion objects is not based on optical flow estimation and no a priori on the scene is taken into account. The algorithm is fully described in [8] and it is based on spatial morphological gradient over image links. The formula is given hereafter and image results are presented in figure 10.

$$mcm^t = \inf(\left|g^{t+1} - g^t\right|, \left|g^t - g^{t-1}\right|)$$

The main advantages of this method is to only memorize three successive images to detect motion parts. However the camera must be fixed not to disturb the detection. If objects to be detected move too slowly regarding the camera frame rate, gradient can not be calculated using t^{-1}, t, t^{+1} images but t^{-n}, t, t^{+n} sequences. The motion detection algorithm could be calculated easily by storing at each step the gradient.

Images must be also filtered with size 5 alternate sequential filters (ASF) before calculating motion detection in order to remove noise due to CMOS sensor.

Op. \ Arch.	P4	Trimedia	Storm-1	Our SoC
mcm	0.7	3.6	0.08	1.14
Asf	13.2	121	0.8	1.52
Total	13.9	124.6	0.88	2.66

Figure 9: Motion detection benchmark (in ms)

Figure 9 shows timing results for each application stage with multiple targeted architectures for 320×240 8 bits images. Our SoC performs the application five times faster than

(a) (b)

Figure 10: Motion detection Example

(a) (b)

Figure 11: License plate extraction

a Pentium 4, but three times slower than a Storm processor running at 700 Mhz. It is worth mentioning our solution could be run at 400 Mhz, thus giving the opportunity to be faster than the Storm processor. Moreover "mcm" could not be fully mapped in the pipeline, it imply to do multiple passes in the pipeline and loosing computing time.

4.3 Licence plate extraction

The method used to extract licence plate is based on the adjacency of characters. A similar method is described in [9]. The algorithm is mainly based on Tophat transformations with structuring element of size equal to characters width. Tophat transformations are described in [2] and are defined as follows :

$$Tophat_{\gamma}(f) = f - \gamma(f)$$

$$Tophat_{\varphi}(f) = \varphi(f) - f$$

Moreover post-filtering after threshold is needed to remove false positive detection using vertical and horizontal openings with various structuring element sizes depending on the image plate scale. Equations are given hereafter and image results are presented in figure 11.

$$g(f) = \inf(f - \gamma(f), \varphi(f) - f)$$

$$plate(f) = \gamma(threshold(g(f), a))$$

Timing performance are similar to those obtained with motion detection application and are given in figure 12.

Op. \ Arch.	P4	Trimedia	Storm-1	Our SoC
g+plate	11,91	103.99	0.65	1.95

Figure 12: Licence plate extraction benchmark (in ms)

5. CONCLUSION AND FUTURE WORK

The Soc described in this paper is a powerful and efficient system for neighbourhood processing such as mathematical morphology and convolution. The peak computational power of one PoC processor is 3,20 Gops at 100 Mhz, but this could be greatly improved by using 64 bits memory words and a higher frequency. For example with 32 bits word size and a 400 Mhz frequency, the computational power peak for one PoC will be 12 Gops, that will bring a 8-stage SPoC to 204,8 Gops. The most important challenge with SPoC is to use the pipeline as much as possible. However, in real application this is not always possible, so our investigation is oriented to the automatic pipeline generation for dynamic partial reconfigurable architecture and automatic software mapping to schedule applications parts efficiently.

REFERENCES

[1] Dan S. Bloomberg (Editor) John Goutsias (Editor), Luc Vincent (Editor). *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2000.

[2] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[3] S.H.D. Hulleman, W. Helbert, H. Chanzy, I. Andreadis, A. Gasteratos, and P. Tsalides. An asic for fast grey-scale dilation. *Microprocessors and Microsystems*, 20:89–95(7), April 1996.

[4] Hugo Hedberg, Fredrik Kristensen, Peter Nilsson, and Viktor wall. A low complexity architecture for binary image erosion and dilation using structuring element decomposition. In *ISCAS (4)*, pages 3431–3434. IEEE, 2005.

[5] Jaromir Brambor. *Algorithmes de la morphologie mathmatique pour les architectures orientes flux*. Thèse de doctorat en morphologie mathmatique, ENSMP, 2006. Dirige par Michel Bilodeau.

[6] B.K. Khailany, T. Williams, J. Lin, E.P. Long, M. Rygh, D.W. Tovey, and W.J. Dally. A programmable 512 gops stream processor for signal, image, and video processing. *Solid-State Circuits, IEEE Journal of*, 43(1):202–213, Jan. 2008.

[7] J-C. Klein and R. Peyrard. Pimm1, an image processing asic based on mathematical morphology. In *Second Annual IEEE ASIC Seminar and Exhibit*, pages 7.1.1–7.1.4, Rochester, September 25-28, 1989, 1989. 1263 CF L-32/90/MM.

[8] L. Biancardini, E. Dokladalova, S. Beucher, and L. Letellier. From moving edges to moving regions. In *IbPRIA 2005, Iberian Conference on Pattern Recognition and Image Analysis*, page 10, Estoril, Portugal, June 7-9 2005, 2005.

[9] Antonio Albiol, J. Manuel Mossi, Alberto Albiol, and Valery Naranjo. Automatic license plate reading using mathematical morphology. In *Proceedings of the The 4th IASTED International Conference on Visualization, Imaging and Image Processing*, Marbella, Spain, september 2004.